

Workshop on the ACTS Toolkit

August 24–27, 2004

National Energy Research Scientific Computing Center

TAO – Toolkit for Advanced Optimization

Steve Benson, Lois Curfman McInnes
Jorge J. Moré, and Jason Sarich

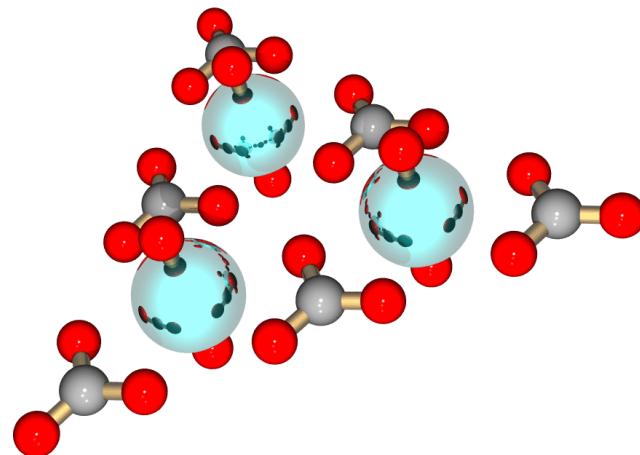
<http://www.mcs.anl.gov/tao>

Mathematics and Computer Science Division
Argonne National Laboratory

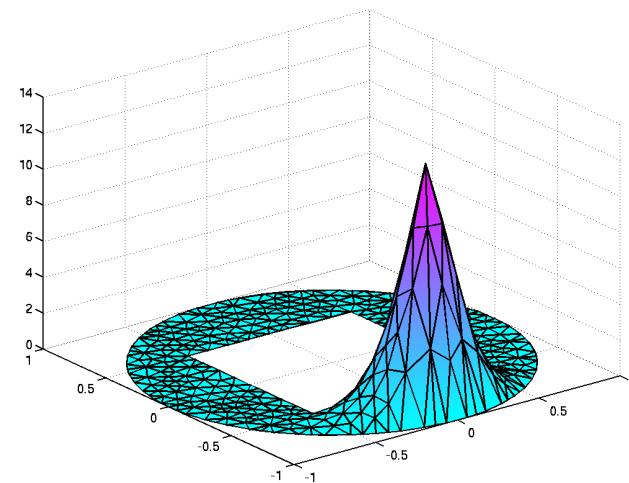


Background: Nonlinearly Constrained Optimization

$$\min \{f(x) : x_l \leq x \leq x_u, c_l \leq c(x) \leq c_u\}$$



Electronic structure optimization



Mesh-based optimization

Note. Image of $(UO_2)_3(CO_3)_6$ courtesy of Wibe deJong (PNNL)

Outline

- ◊ TAO - Recent Developments
- ◊ Optimization applications
- ◊ TAO algorithms
- ◊ Performance for Grid Sequencing – Stalking optimality
- ◊ Using TAO

TAO: Toolkit for Advanced Optimization

The process of nature by which all things change and
which is to be followed for a life of harmony.

- ◊ Object-oriented techniques
- ◊ Component-based interaction
- ◊ Leverage of existing parallel computing infrastructure
- ◊ Reuse of external toolkits (linear solvers, preconditioners, . . .)

TAO: Recent Developments

- ◊ Version 1.7 (August 2004)
- ◊ Source code, documentation, tutorials, example problems, . . .
- ◊ Development of BLMVM
- ◊ TAO components: MPQC (Sandia) and NWChem (PNNL)
- ◊ Grid sequencing via Distributed Arrays (PETSc)
- ◊ Gradients of grid functions via ADIC

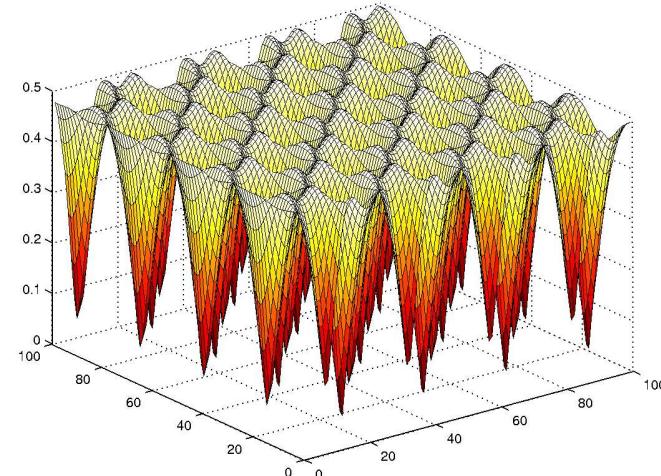
Powered by PETSc and ADIC!

The Ginzburg-Landau Model for Superconductivity

Minimize the Gibbs free energy

$$\int_{\mathcal{D}} \left\{ -|v(x)|^2 + \frac{1}{2} |v(x)|^4 + \|[\nabla - iA(x)] v(x)\|^2 + \kappa^2 \|(\nabla \times A)(x)\|^2 \right\} dx$$

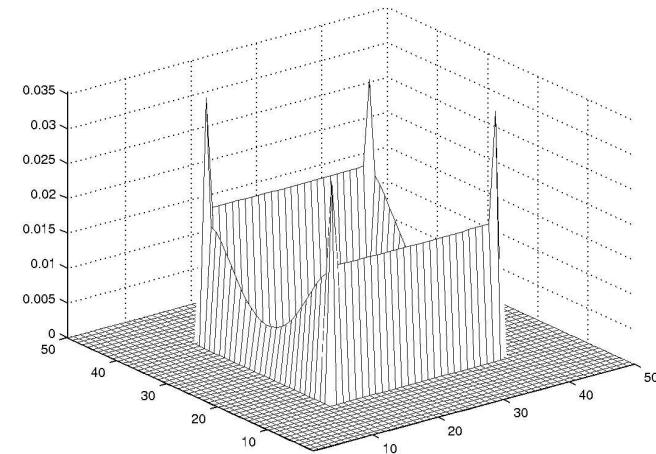
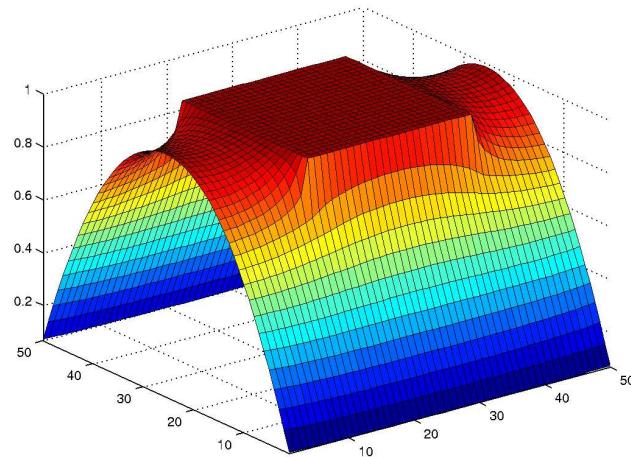
Order parameter $v : \mathbb{R}^2 \rightarrow \mathbb{C}$
Vector potential $A : \mathbb{R}^2 \rightarrow \mathbb{R}^2$



Non-convex function. Hessian is singular. Unique minimizer.

Minimal Surface with Obstacles

$$\min \left\{ \int_{\mathcal{D}} \sqrt{1 + \|\nabla v(x)\|^2} dx : v \geq v_L \right\}$$



Number of active constraints depends on the height of the obstacle.
The solution $v \notin C^1$. Almost all multipliers are zero.

Isomerization of α -pinene

Determine the reaction coefficients by minimizing

$$\sum_{j=1}^8 \|y(\tau_j; \theta) - z_j\|^2,$$

where z_j are the measurements and

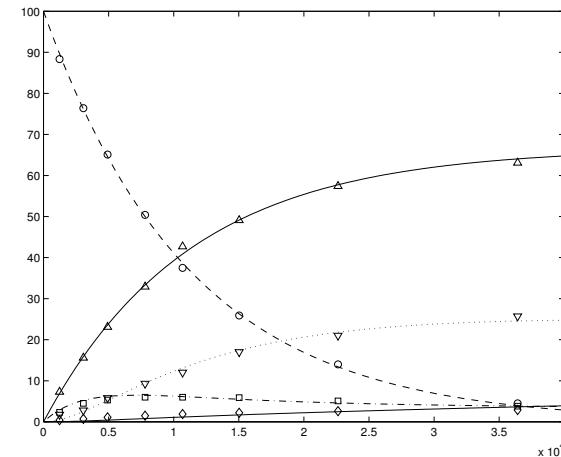
$$y'_1 = -(\theta_1 + \theta_2)y_1$$

$$y'_2 = \theta_1 y_1$$

$$y'_3 = \theta_2 y_1 - (\theta_3 + \theta_4)y_3 + \theta_5 y_5$$

$$y'_4 = \theta_3 y_3$$

$$y'_5 = \theta_4 y_3 - \theta_5 y_5$$



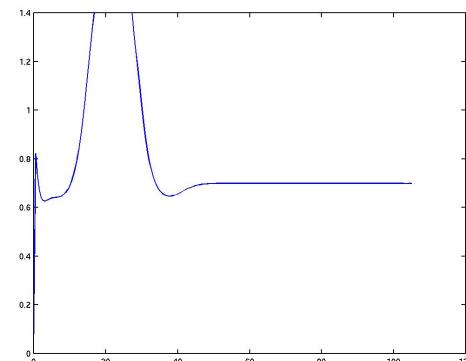
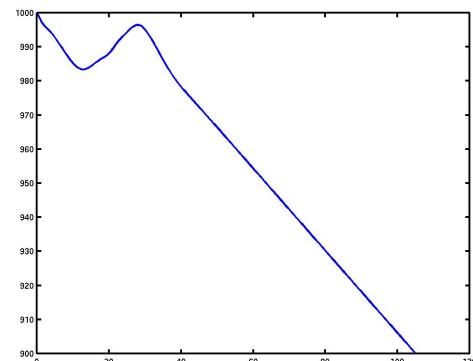
Hang Glider

Maximize the final position of a glider whose flight has a thermal updraft at 250 meters. The equations of motion are

$$x'' = \frac{1}{m}(-L \sin(\eta) - D \cos(\eta)), \quad y'' = \frac{1}{m}(L \cos(\eta) - D \sin(\eta)) - g,$$

where (x, y) is the position, m is the mass, and

$$\begin{aligned}\eta &= \eta(x, x', y') \\ L &= L(x, x', y', c_L) \\ D &= D(x, x', y', c_L) \\ 0 \leq c_L(t) &\leq c_M\end{aligned}$$



The control (drag) c_L is not differentiable at the switching points.

Transition States

Given a continuously differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ and two points x_a and x_b , determine a critical point x^* on a minimal energy path between x_a and x_b .

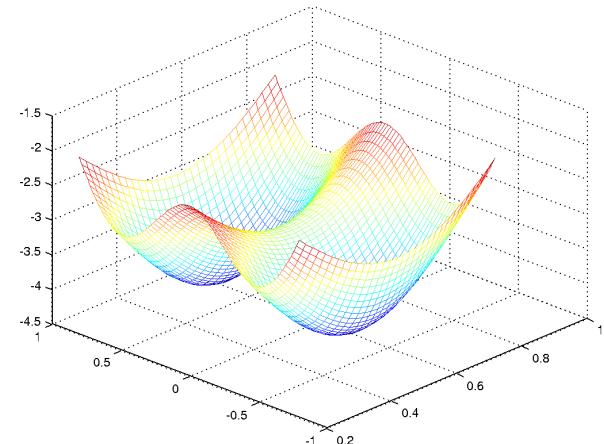
- A fundamental problem in biology, chemistry, and mathematics

A critical value of f is

$$\gamma = \inf_{p \in \Gamma} \{\max \{f[p(t)] : t \in [0, 1]\}\}$$

when Γ is defined by

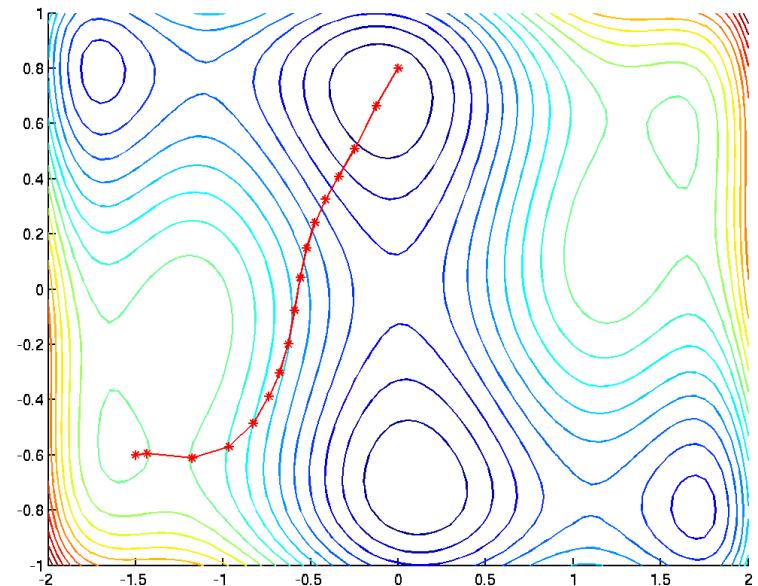
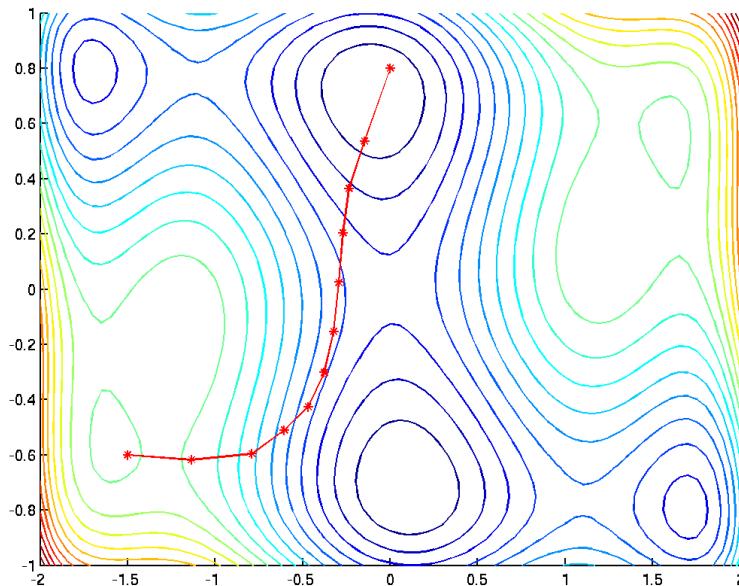
$$\Gamma = \{p \in C[0, 1] : p(0) = x_a, p(1) = x_b\}.$$



The Elastic String Algorithm

Compute breakpoints $x_k \in \mathbb{R}^n$ for a piecewise linear path such that

$$\min \left\{ \max \{f(x_1), \dots, f(x_m)\} : \|x_{k+1} - x_k\| \leq h_k, \ 0 \leq k \leq m \right\}.$$

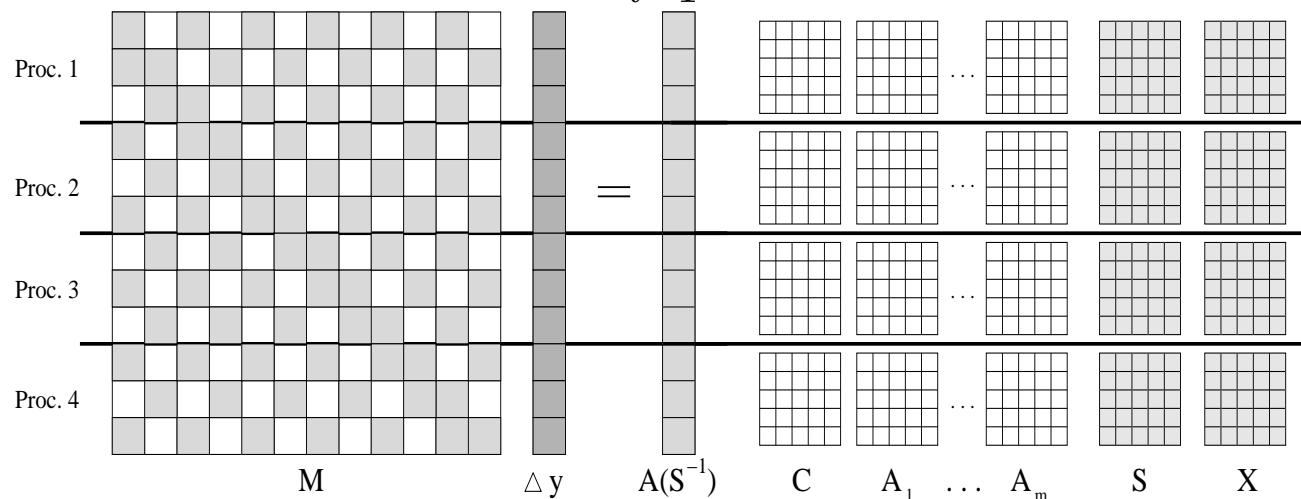


Matrix Inequalities

Applications in

- ◇ control theory
- ◇ combinatorial relaxations

have constraints of the form: $\sum_{i=1}^m A_i y_i - C \succeq 0$,



TAO Algorithms for Bound-Constrained Optimization

$$\min \{f(x) : x_l \leq x \leq x_u\}$$

- ◊ Conjugate gradient algorithms
- ◊ Limited-memory variable-metric algorithms
- ◊ Newton algorithms

You must supply the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ and the gradient

$$\nabla f(x) = (\partial_i f(x))$$

For Newton methods you also need to supply the Hessian matrix.

$$\nabla^2 f(x) = (\partial_{i,j} f(x))$$

Conjugate Gradient Algorithms

$$x_{k+1} = x_k + \alpha_k p_k$$

$$p_{k+1} = -\nabla f(x_k) + \beta_k p_k$$

where α_k is determined by a line search.

Three choices of β_k are possible ($g_k = \nabla f(x_k)$):

$$\beta_k^{FR} = \left(\frac{\|g_{k+1}\|}{\|g_k\|} \right)^2, \quad \text{Fletcher-Reeves}$$

$$\beta_k^{PR} = \frac{\langle g_{k+1}, g_{k+1} - g_k \rangle}{\|g_k\|^2}, \quad \text{Polak-Rivièvre}$$

$$\beta_k^{PR+} = \max \{ \beta_k^{PR}, 0 \}, \quad \text{PR-plus}$$

Limited-Memory Variable-Metric Algorithms

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$$

where α_k is determined by a line search.

The matrix H_k is defined in terms of information gathered during the previous m iterations.

- ◊ H_k is positive definite.
- ◊ Storage of H_k requires $2mn$ locations.
- ◊ Computation of $H_k \nabla f(x_k)$ costs $(8m + 1)n$ flops.

Trust Region Newton Algorithm

At each iteration the step s_k (approximately) minimizes

$$\min \{q_k(x_k + s) : s_i = 0, i \in \mathcal{A}_k, x_l \leq x_k + s \leq x_u, \|s\| \leq \Delta_k\}$$

where q_k is the quadratic approximation,

$$q_k(w) = \langle \nabla f(x_k), w \rangle + \frac{1}{2} \langle w, \nabla^2 f(x_k) w \rangle,$$

to the function, and Δ_k is the trust region bound.

- ◇ Predict an active set \mathcal{A}_k .
- ◇ Compute a step s_k
- ◇ $x_{k+1} = x_k + s_k$ if $f(x_k + s_k) < f(x_k)$, otherwise $x_{k+1} = x_k$.
- ◇ Update Δ_k .

Mesh Sequencing Issues

- How much does mesh-sequencing save?
- Does mesh-sequencing resolve the convergence issue?
 - ◊ What is the order of convergence of $f(x_h^*)$?
 - ◊ What is the order of convergence of x_h^* ?
 - ◊ How does the number of active constraints at x_h change?
 - ◊ What tolerance do we use to obtain x_h ?
 - ◊ What is the impact on iterative methods on these results?

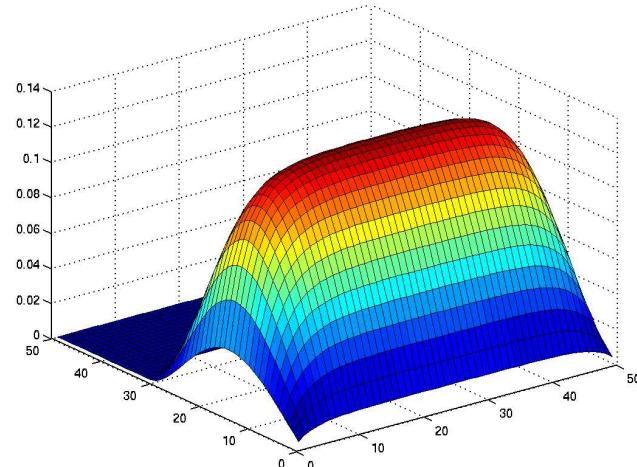
TAO Benchmark: Pressure in a Journal Bearing

$$\min \left\{ \int_{\mathcal{D}} \left\{ \frac{1}{2} w_q(x) \|\nabla v(x)\|^2 - w_l(x)v(x) \right\} dx : v \geq 0 \right\}$$

$$w_q(\xi_1, \xi_2) = (1 + \epsilon \cos \xi_1)^3$$

$$w_l(\xi_1, \xi_2) = \epsilon \sin \xi_1$$

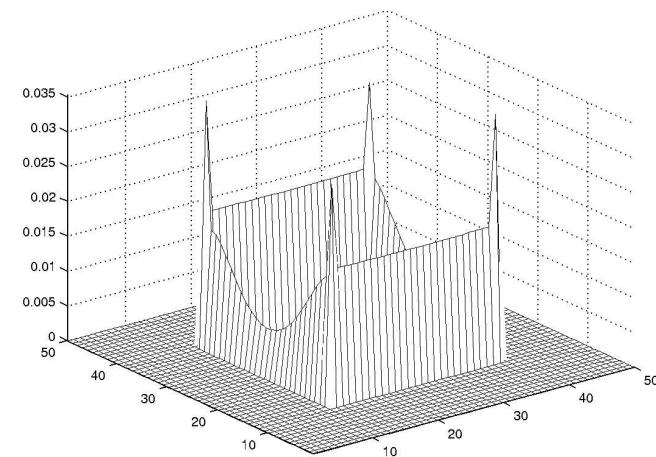
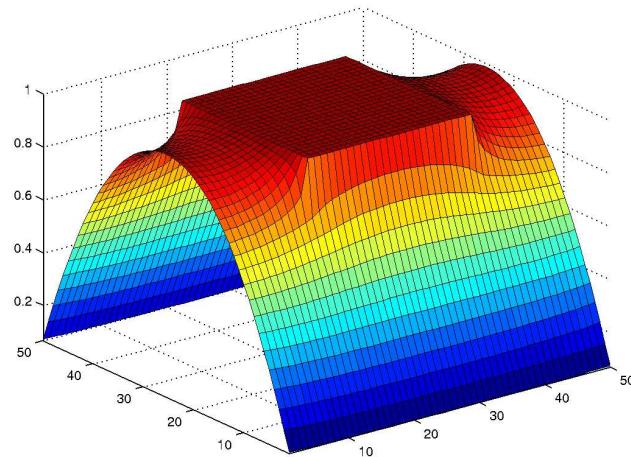
$$\mathcal{D} = (0, 2\pi) \times (0, 2b)$$



Number of active constraints depends on the choice of ϵ in $(0, 1)$.
Nearly degenerate problem. Solution $v \notin C^2$.

TAO Benchmark: Minimal Surface with Obstacles

$$\min \left\{ \int_{\mathcal{D}} \sqrt{1 + \|\nabla v(x)\|^2} dx : v \geq v_L \right\}$$



Number of active constraints depends on the height of the obstacle.
The solution $v \notin C^1$. Almost all multipliers are zero.

Mesh-Sequencing Performance: Journal Bearing Problem

	Number of processors					
	16	32	64			
Mesh	田	□	田	□	田	□
769 × 769	17	283	10	142	7	86
1537 × 1537	73	3751	40	1861	22	938

	Mesh	16	32	64
Improvements	769 × 769	16	14	12
	1537 × 1537	51	46	42

Mesh-Sequencing Performance: Obstacle Problem

	Number of processors					
	16		32		64	
Mesh	田	□	田	□	田	□
769 × 769	69	†	37	†	24	†
1537 × 1537	444	†	235	†	121	†

† No convergence after 500 iterations!

Using TAO

There are three parts of a TAO application program:

- ◊ **Vector** and **matrix** objects from PETSc or other numerical packages.
- ◊ **Applications** contain routines that evaluate an objective function, define constraints on the variables, and provide derivative information.
- ◊ The **TAO solver** with desired algorithmic options and tolerances.

Using TAO with PETSc

```
TAO_SOLVER      tao;          /* TAO Optimization solver      */
TAO_APPLICATION app;          /* TAO Application using PETSc   */
AppCtx          user;          /* user-defined application context */
Vec             x;            /* solution vector               */
Mat             H;            /* Hessian Matrix                */

VecCreateSeq(PETSC_COMM_SELF,n,&x);
MatCreateSeqAIJ(PETSC_COMM_SELF,n,n,nz,PETSC_NULL,&H);
TaoCreate(PETSC_COMM_SELF,'tao_lmvm',&tao);
TaoApplicationCreate(PETSC_COMM_SELF,&app);
TaoAppSetInitialSolutionVec(app,x);
TaoAppSetObjectiveRoutine(app, FormFunction,(void *)&user);
TaoAppSetGradientRoutine(app,FormGradient,(void *)&user);
TaoAppSetHessianMat(app,H,H);
TaoAppSetHessianRoutine(app,FormHessian,(void *)&user);
TaoSolveApplication(app,tao);
VecView(x,PETSC_VIEWER_STDOUT_SELF);
```

Objective Function and Gradient Evaluation

```
typedef struct {           /* Used in the minimum surface area problem */
    int      mx, my;        /* discretization in x, y directions */
    int      bmx, bmy, bheight;   /* The size of the plate */
    double   bheight;        /* The height of the plate */
    double   *bottom, *top, *left, *right; /* boundary values */
} AppCtx;

int FormFunction(TAO_APPLICATION app, Vec x, double *fcn, void *userCtx){
    AppCtx *user = (AppCtx *)userCtx;
    ...
}

int FormGradient(TAO_APPLICATION app, Vec x, Vec g, void *userCtx){
    AppCtx *user = (AppCtx *)userCtx;
    ...
}

int FormHessian(TAO_APPLICATION app, Vec x, Mat *H, Mat *H, int *flag, void *userCtx){
    AppCtx *user = (AppCtx *)userCtx;
    ...
}
```

Creating and Using a TAO Application

```
TAO_SOLVER      tao;          /* TAO Optimization solver      */
TAO_APPLICATION app;          /* TAO Application using PETSc   */
AppCtx          user;          /* user-defined application context */
Vec             x;            /* solution vector               */
Mat             H;            /* Hessian Matrix                */

VecCreateSeq(PETSC_COMM_SELF,n,&x);
MatCreateSeqAIJ(PETSC_COMM_SELF,n,n,nz,PETSC_NULL,&H);
TaoCreate(PETSC_COMM_SELF,"tao_lmvm",&tao);
TaoApplicationCreate(PETSC_COMM_SELF,&app);
TaoAppSetInitialSolutionVec(app,x);
TaoAppSetObjectiveRoutine(app,FormFunction,(void *)&user);
TaoAppSetGradientRoutine(app,FormGradient,(void *)&user);
TaoAppSetHessianMat(app,H,H);
TaoAppSetHessianRoutine(app,FormHessian,(void *)&user);
TaoSolveApplication(app,tao);
VecView(x,PETSC_VIEWER_STDOUT_SELF);
```

Creating and Using a TAO Solver

```
TAO_SOLVER      tao;          /* TAO Optimization solver      */
TAO_APPLICATION app;          /* TAO Application using PETSc   */
AppCtx          user;          /* user-defined application context */
Vec             x;            /* solution vector               */
Mat             H;            /* Hessian Matrix                */

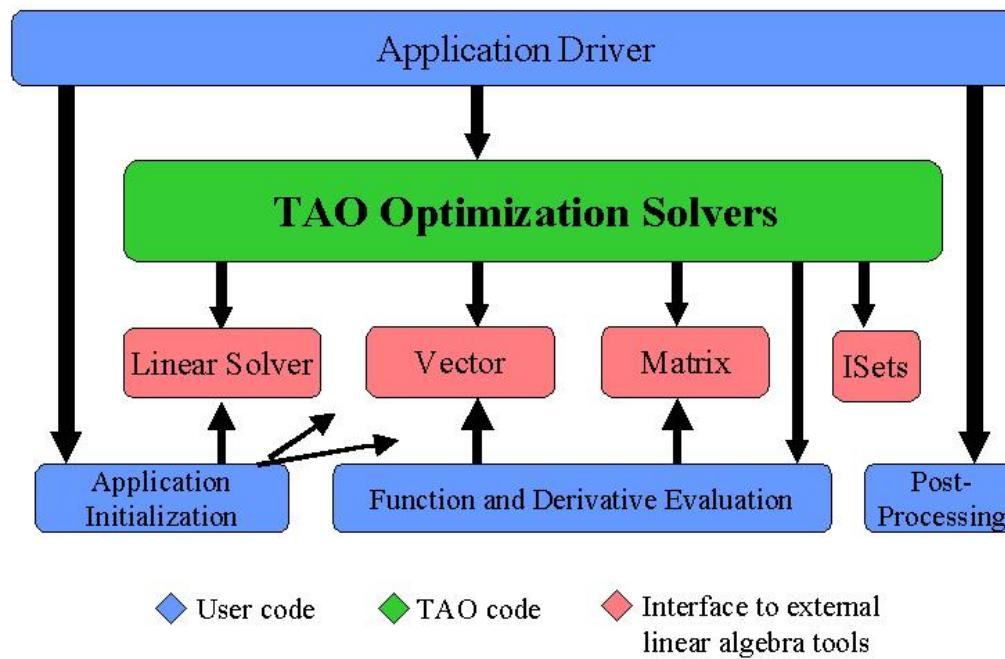
VecCreateSeq(PETSC_COMM_SELF,n,&x);
MatCreateSeqAIJ(PETSC_COMM_SELF,n,n,nz,PETSC_NULL,&H);
TaoCreate(PETSC_COMM_SELF,"tao_lmvm",&tao);
TaoApplicationCreate(PETSC_COMM_SELF,&app);
TaoAppSetInitialSolutionVec(app,x);
TaoAppSetObjectiveRoutine(app,FormFunction,(void *)&user);
TaoAppSetGradientRoutine(app,FormGradient,(void *)&user);
TaoAppSetHessianMat(app,H,H);
TaoAppSetHessianRoutine(app,FormHessian,(void *)&user);
TaoSolveApplication(app,tao);
VecView(x,PETSC_VIEWER_STDOUT_SELF);
```

A TAO Program Outline

```
TAO_SOLVER      tao;          /* TAO Optimization solver      */
TAO_APPLICATION app;          /* TAO Application using PETSc   */
AppCtx          user;          /* user-defined application context */
Vec             x;              /* solution vector               */
Mat             H;              /* Hessian Matrix                */

VecCreateSeq(PETSC_COMM_SELF,n,&x);
MatCreateSeqAIJ(PETSC_COMM_SELF,n,n,nz,PETSC_NULL,&H);
TaoCreate(PETSC_COMM_SELF,"tao_lmvm",&tao);
TaoApplicationCreate(PETSC_COMM_SELF,&app);
TaoAppSetInitialSolutionVec(app,x);
TaoAppSetObjectiveRoutine(app,FormFunction,(void *)&user);
TaoAppSetGradientRoutine(app,FormGradient,(void *)&user);
TaoAppSetHessianMat(app,H,H);
TaoAppSetHessianRoutine(app,FormHessian,(void *)&user);
TaoSolveApplication(app,tao);
VecView(x,PETSC_VIEWER_STDOUT_SELF);
```

TAO Application Programs



Using PETSc Objects on Multiple Processors

```
TAO_SOLVER      tao;          /* TAO Optimization solver      */
TAO_APPLICATION app;          /* TAO Application using PETSc   */
AppCtx          user;          /* user-defined application context */
Vec             x;            /* solution vector               */
Mat             H;            /* Hessian Matrix                */

VecCreateMPI(PETSC_COMM_WORLD,n,&x);
MatCreateMPIAIJ(PETSC_COMM_WORLD,nlocal,nlocal,n,n,d_nz,d_nnz,o_nz,o_nnz,&H);
TaoCreate(PETSC_COMM_WORLD,"tao_lmvm",&tao);
TaoApplicationCreate(PETSC_COMM_WORLD,&app);
TaoAppSetInitialSolutionVec(app,x);
TaoAppSetObjectiveRoutine(app,FormFunction,(void *)&user);
TaoAppSetGradientRoutine(app,FormGradient,(void *)&user);
TaoAppSetHessianMat(app,H,H);
TaoAppSetHessianRoutine(app,FormHessian,(void *)&user);
TaoSolveApplication(app,tao);
VecView(x,PETSC_VIEWER_STDOUT_WORLD);
```

Convergence

Absolute tolerances specify acceptable errors in the optimality of the function and the constraints.

$$f(x) \leq f(x^*) + \epsilon_{fatol}$$

Relative tolerances specify the number of significant digits required in the solution and the constraints.

$$f(x) \leq f(x^*) + \epsilon_{frtol} |f(x^*)|$$

These tolerance can be changed

```
int TaoSetTolerances(TAO_SOLVER solver,double fatol,double frtol,  
                     double catol,double crtol)
```

TAO Basic Facilities

- ◊ TaoAppSetInitialSolutionVec
- ◊ TaoAppSetVariableBounds
- ◊ TaoGetLinearSolver
- ◊ TaoFromOptions
- ◊ TaoAppSetMonitor
- ◊ TaoView
- ◊ ...